BAB 2 LANDASAN KEPUSTAKAAN

Bab ini berisi uraian mengenai dasar teori yang berkaitan dengan penelitian ini, serta kajian pustaka dari penelitian-penelitian terdahulu yang dapat mendukung penelitian ini.

2.1 Kajian Pustaka

Tabel 2.1 di bawah ini merupakan beberapa kajian pustaka dari hasil penelitian sebelumnya yang terkait dengan penelitian yang dilakukan oleh penulis saat ini.

Tabel 2.1 Kajian Pustaka

No	Nama Penulis, Tahun dan Judul	Persamaan	Perbedaan	
			Penelitian terdahulu	Rencana penelitian
1.	Veronika Abramova, Jorge Bernardino, Pedro Furtado. (2014, *September*). *Testing Cloud Benchmark Scalability with Apache Cassandra*.	Melakukan pengujian dengan aplikasi *YCSB*.	Hanya melakukan pengujian performa pada *Cassandra*.	Melakukan pengujian performa pada *HBase dan Cassandra*.
2.	Surya Narayanan Swaminathan, Ramez Elmasri. (2016, *October*). *Quantitative Analysis of Scalable NoSQL Databases*.	Melakukan pengujian performa pada *HBase dan Cassandra*.	Hanya melakukan pengujian dengan aplikasi *YCSB* dan tidak menarik kesimpulan *database* mana yang terbaik.	Melakukan pengujian dengan aplikasi *YCSB* & *JMeter*, serta menarik kesimpulan tentang *database* mana yang terbaik.
3.	Jörn Kuhlenkamp, Markus Klems, Oliver Röss. (2014, *August*). *Benchmarking Scalability and Elasticity of Distributed Database Systems*.	Melakukan pengujian performa pada *HBase dan Cassandra*.	Hanya melakukan pengujian dengan aplikasi *YCSB*.	Melakukan pengujian dengan aplikasi *YCSB* dan *JMeter*.

Sumber: (Abramova, Bernardino and Furtado, 2014) (Swaminathan and Elmasri, 2016) (Kuhlenkamp, Klems and Röss, 2014)

2.2 *Database*

Database adalah kumpulan data. Data merupakan fakta yang diketahui, dapat direkam dan memiliki makna tersirat. Sebuah *Database* biasanya merepresentasikan beberapa objek asli dari dunia nyata dan digunakan untuk tujuan tertentu oleh satu atau lebih kelompok pengguna. Untuk menerapkan dan memelihara suatu *database* secara terkomputerisasi, pengguna dapat menggunakan suatu *Database Management System* (*DBMS*). *DBMS* adalah perangkat lunak yang berfungsi untuk membuat dan memanajemen suatu *database*.

Sebagai contoh, nama, nomor telepon, dan alamat dari orang yang kita kenal. Data-data tersebut biasanya disimpan dalam sebuah buku alamat berindeks atau mungkin disimpan pada *hard drive*, menggunakan komputer pribadi dan perangkat lunak seperti *Microsoft Access* atau *Excel*. Koleksi data dengan makna implisit ini bisa disebut dengan *database*, sedangkan perangkat lunak seperti *Microsoft Access* dan *Excel* ini bisa disebut sebagai *DBMS* (Elmasri and Navathe, 2011).

2.3 *Not Only SQL*

Database Not Only SQL (NoSQL) merupakan model baru *database* non-relasional, yang pada perkembangannya menarik perhatian masyarakat penelitian dan perusahaan dengan cepat. *NoSQL Database* dapat menangani semua tuntutan penyimpanan dan pengaksesan data pada suatu sistem perusahaan / aplikasi web yang berbeda-beda dengan cepat dan efisien. *NoSQL Database* memiliki fitur tambahan yang membedakannya dengan *Relational Database Management Systems (RDBMS)*, seperti tidak membutuhkan skema *database* kaku untuk didefinisikan dan memiliki skalabilitas horizontal yang mudah.

NoSQL Database dirancang untuk menyediakan skalibilitas *shared-nothing* horizontal yang ditunjukan dengan distribusi data melalui *server* yang berbeda. Dengan demikian, *NoSQL Database* dikembangkan untuk dapat melakukan skalabilitas (penambahan beban) dengan mudah dan dapat mendistribusikan data dengan efisien. Ketika jumlah *server* meningkat, sistem ini justru mampu melayani permintaan dengan lebih efisien. Semua permintaan dieksekusi secara paralel, sehingga menghasilkan *Throughput* yang lebih tinggi dan waktu eksekusi *query* yang lebih rendah (Abramova, Bernardino and Furtado, 2014).

Ada 4 tipe *NoSQL Database* berdasarkan model penyimpanannya, yaitu:

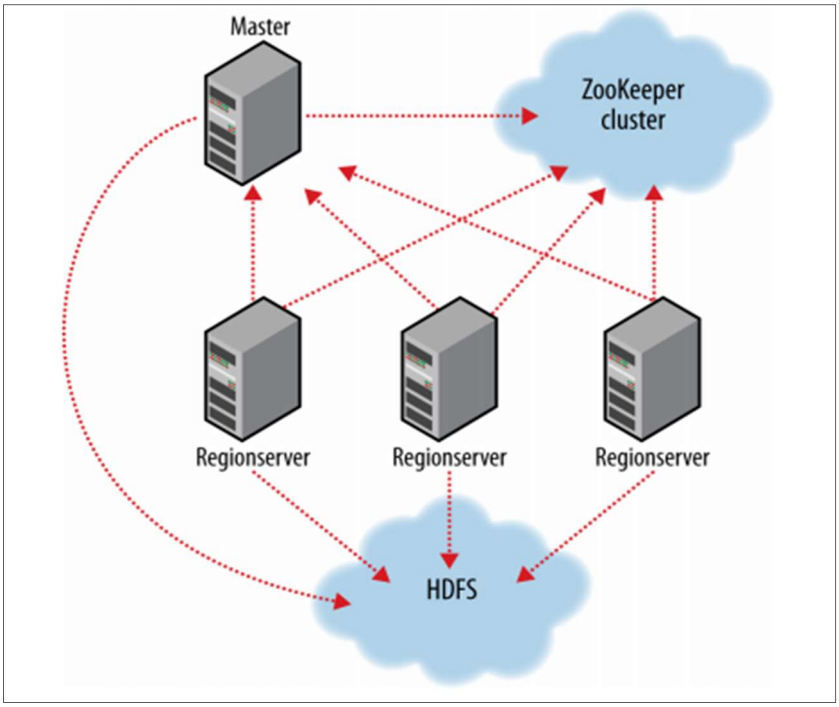
1. *Key-Value based*, data disimpan dalam bentu kunci-isi berpasangan, maksudnya pada tipe ini terdapat dua bagian bagian yaitu sebuah string yang mempresentasikan sebuah kunci (*key*) dan data aktual yang merupakan nilai (*value*) sehinga akan membentuk suatu pasangan kunci-isi (*key-value*). Contoh *database* yang menggunakan konsep *key-value* based yaitu *DynamoDB* dan *Aerospike*.

2. *Column-oriented based*, data disimpan dalam kolom-kolom, walaupun menggunakan konsep kolom, tetapi tipe ini tidak seperti konsep relasional kolom *database* pada umumnya. Data tidak disimpan dalam tabel tetapi disimpan dalam suatu arsitektur distribusi yang sangat besar. Pada tipe ini, setiap *key* dihubungkan dengan satu atau lebih atribut (kolom). Contoh *database* yang menggunakan konsep *Column-oriented* adalah *HBase* dan *Cassandra*.
3. *Document-store based*, data disimpan dalam dalam satu bentuk dokumen. Dokumen pada *document-store database* sama dengan *record* pada *relational database*. Dokumen yang disimpan dapat berupa format *PDF*, *XML*, *JSON* dan lain-lain. Dokumen sendiri bisa terdiri dari *key-value, unique key* digunakan untuk mempresentasikan dokumen. *Key* bisa berupa sebuah *string* sederhana atau sebuah *string* yang mengarahkan ke sebuah *URI* atau *path*. Contoh *database* yang menggunakan konsep *document-store* adalah *CouchDB* dan *MongoDB*.
4. *Graph based*, penyimpanan data dilakukan dalam bentuk graph. Sebuah graph terdiri dari *node* dan *edge*. *Node* menggambarkan sebuah objek dan *edge* menggambarkan sebuah relasi antar objek. Setiap *node* memiliki *direct pointer*, titik tersebut merupakan *node* yang berdekatan. *Database Graph* memiliki fitur *schema less* dan penyimpanan efisien pada data yang semi terstruktur. Contoh *database* yang menggunakan konsep *graph* adalah *Neo4j*.
5. *Object oriented based*, data disimpan dan direpresentasikan sebagai sebuah objek. Tipe ini menggabungkan antara konsep *Object Oriented Programming (OOP)* dengan dasar *database*. Objek data disimpan dengan fitur yang sama dengan *OOP* seperti *encapsulation, polymorphism* and *inheritance*. Kelas dapat disamakan dengan tabel. Setiap objek mempunyai sebuah *identifier* untuk membedakan objek satu dengan objek yang lainnya. Contoh *database* yang menggunakan konsep ini adalah *db4o* (Nayak, Poriya and Poojary, 2013).

2.4 *Apache HBase*

Apache HBase adalah sebuah proyek *open-source* yang mengimplementasikan desain sistem dan mesin penyimpanan *Big-Table*. Arsitektur *HBase* terdiri dari satu *master server* aktif dan beberapa *slave servers* yang disebut sebagai *region servers*. Tabel dibagi pada setiap *region* dan didistribusikan ke *region servers*. Sebuah *region* memegang berbagai baris untuk kelompok *sharding* tunggal, yaitu, *column family*. *HBase master* bertanggung jawab untuk daerah pemetaan tabel ke *region servers*.

HBase umumnya digunakan dalam kombinasi dengan *Hadoop Distributed File System (HDFS)* yang digunakan untuk penyimpanan data dan replikasi. Arsitektur *HDFS* mirip dengan arsitektur *HBase*. HDFS memiliki satu *master server*, *name node*, dan beberapa *data node*. *Name node* yang mengelola *file meta data* sistem, sedangkan *data node* yang menyimpan data yang sebenarnya (Kuhlenkamp, Klems and Röss, 2014).



Gambar 2.1 Arsitektur *Apache HBase*
Sumber: (White, 2015)

Gambar 2.1 di atas merupakan gambaran umum dari arsitektur *HBase*. Gambar tersebut menjelaskan bahwa pada suatu sistem arsitektur *HBase* harus terdiri dari satu *node Master HBase* dan satu atau lebih *node Regionserver (Slave nodes)*. *Node Master HBase* ini bertugas untuk mengatur *cluster* dari satu atau lebih *node Regionserver* tadi. Gambar di atas merupakan sebuah arsitektur *HBase Distributed mode* yang didalamnya terdapat sebuah *Zookeeper Cluster* yang terdiri dari 1 *node Master* dan 3 *node Regionserver* yang semuanya berjalan di atas *HDFS*.

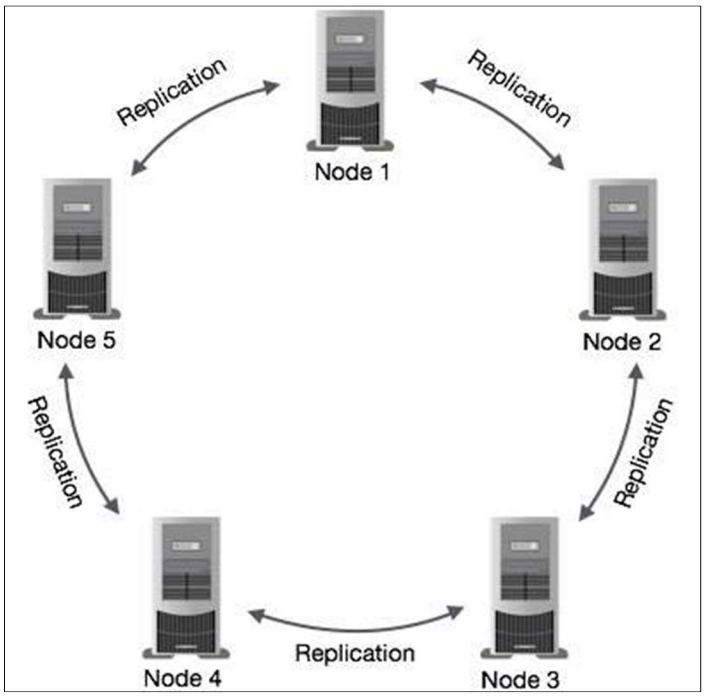
Suatu arsitektur *HBase* bisa berupa *Standalone mode* (hanya menggunakan satu *node* yang berfungsi sebagai *Master* sekaligus *Slaves*) ataupun berupa *Distributed mode* (menggunakan dua *Slave nodes* atau lebih). Untuk memanajemen *cluster* dari *Slaves* pada arsitektur *HBase* dengan *Distributed mode* digunakan suatu perangkat lunak bernama *Apache Zookeeper.*

Apache Zookeeper adalah layanan koordinasi terdistribusi *open-source* untuk aplikasi terdistribusi. Aplikasi ini digunakan untuk menerapkan layanan sinkronisasi, pemeliharaan konfigurasi, serta pengelompokan dan penamaan sistem terdistribusi yang lebih tinggi. Aplikasi ini dirancang agar mudah untuk digunakan. Aplikasi ini berjalan di *Java* dan memiliki *binding* untuk *Java* dan *C* (Zookeeper, 2017).

2.5 *Apache Cassandra*

Apache Cassandra adalah sistem *database* desentralisasi terdistribusi yang awalnya dikembangkan sebagai *write-efficient database* untuk kotak masuk pesan *Facebook*. Perangkat lunak ini dirilis sebagai proyek perangkat lunak *open-source* yang sekarang dikelola oleh *Apache Software Foundation*.

Cassandra mengimplementasikan arsitektur replikasi *Dynamo* serta model data dan mesin penyimpanan *Big-Table* yang dioptimalkan untuk *Throughput* penulisan data yang tinggi dan menyimpan baris dengan jumlah kolom yang besar dalam skema yang fleksibel (Kuhlenkamp, Klems and Röss, 2014).



Gambar 2.2 Arsitektur *Apache Cassandra*
Sumber: (Tutorialspoint, 2017)

Gambar 2.2 di atas merupakan gambaran umum arsitektur *Cassandra*. *Cassandra* menggunakan *Gossip Protocol* pada *background* untuk komunikasi antara *node* satu dengan *node* yang lain dan untuk mendeteksi setiap *node* yang cacat/*down* pada *cluster*. Selain itu, *Gossip Protocol* tersebut juga digunakan untuk mengirimkan hasil replikasi data antara *node-node* yang terdaftar sebagai *seeds* dalam suatu arsitektur *Cassandra*. Dalam gambar di atas yang berperan sebagai *seeds* adalah Node 1, Node 2, Node 3, Node 4 dan Node 5.

2.6 *Internet of Things*

Internet of Things (IoT) merupakan konsep yang memiliki tujuan untuk meningkatkan dan memperluas manfaat dari sebuah konektivitas internet yang tersambung terus-menerus sehingga dapat merambah pada setiap aspek kehidupan manusia. *IoT* menghubungkan benda fisik dan virtual melalui kemampuan komunikasi.

IoT bisa disebut sebagai sebuah infrastruktur jaringan global. Infrastruktur terdiri dari jaringan yang telah ada dan internet berikut pengembangan jaringannya. *IoT* adalah ketika internet dan jaringan memperluas lingkupnya hingga ke tempat-tempat seperti lantai manufaktur, jaringan energi, fasilitas kesehatan, dan transportasi (Cisco, 2017).

2.7 *The Yahoo! Cloud Serving Benchmark*

The Yahoo! Cloud Serving Benchmark (YCSB) adalah alat *benchmarking* modular dengan *adapter* untuk sejumlah sistem *database* terdistribusi. *YCSB* bisa digunakan untuk melakukan *benchmarking* pada sistem *database* terdistribusi seperti *HBase*, *Cassandra*, *Riak*, *MongoDB*, dan banyak lagi. *YCSB* menyediakan paket inti dengan 5 *benchmark workload* A-E.

Sebuah *workload* standar *YCSB* memiliki sebuah kunci baris dan 10 kolom, yang masing-masing diisi dengan 100 *byte* teks *ASCII* acak. Dengan demikian, ukuran standar satu baris adalah 1KB, termasuk ukuran kunci baris. Selain *workload* standar, *YCSB* juga menyediakan *workload template* yang dapat dikonfigurasi sendiri sesuai dengan kebutuhan pengujian (Kuhlenkamp, Klems and Röss, 2014).

2.8 *Apache JMeter*

Apache JMeter adalah sebuah perangkat lunak *open-source* yang dikembangkan oleh *Apache Software Foundation*. *JMeter* merupakan aplikasi berbasis *Java* yang dirancang untuk melakukan tes fungsional dan mengukur kinerja suatu sistem.

Pada awalnya, aplikasi ini dirancang untuk pengujian Aplikasi Web. Namun saat ini sudah diperluas untuk menguji fungsi lainnya. Selain itu, juga terdapat banyak pengembang dari pihak ketiga yang ikut berkontribusi membuat *plugin-plugin* tambahan yang dapat digunakan untuk menguji kinerja suatu sistem/aplikasi menggunakan *JMeter* ini (*JMeter*, 2017).

2.9 Parameter Pengujian

Penelitian ini menggunakan beberapa parameter pengujian untuk menguji performa *HBase* dan *Cassandra*. Adapun parameter uji yang digunakan antara lain:

- *Throughput* atau kecepatan eksekusi *query*, merupakan jumlah operasi yang dieksekusi database tiap detik (Abramova, Bernardino and Furtado, 2014).
- *Latency* atau bisa disebut sebagai *response time*, merupakan lama waktu yang dibutuhkan *database* untuk menanggapi suatu *request* (Abramova, Bernardino and Furtado, 2014).
- *Runtime* atau bisa disebut juga sebagai *Execution Time*, merupakan waktu eksekusi *query* suatu *database* (Abramova, Bernardino and Furtado, 2014).
- *Memory Usage* merupakan persentase kapasitas *memory* (*RAM*) yang dipakai oleh *database server* ketika mengeksekusi suatu operasi transaksi data (misal *insert*) (Pokhilko, 2017).
- *CPU Usage* merupakan persentase kapasitas prosesor (*CPU*) yang dipakai oleh *database server* ketika mengeksekusi suatu operasi transaksi data (misal *insert*) (Pokhilko, 2017).